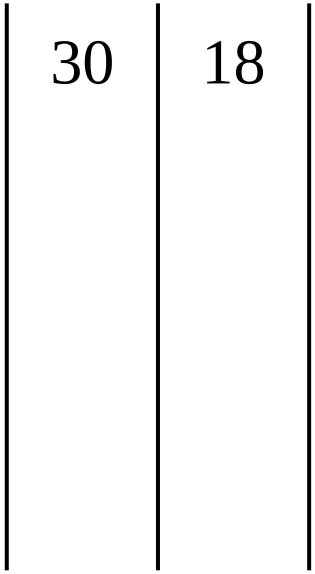
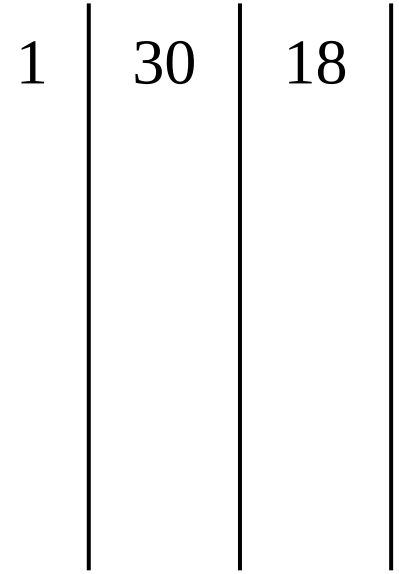
輾轉相除法 (Euclidean Algorithm)



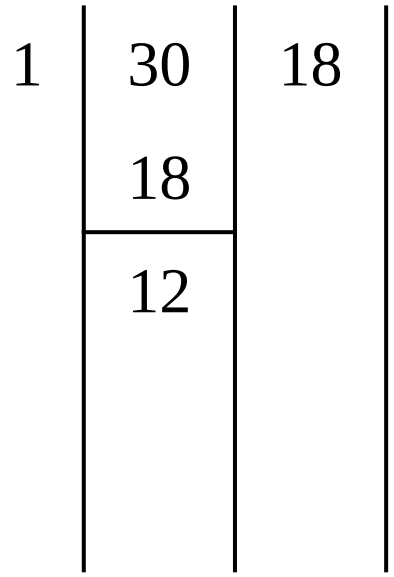
輾轉相除法 (Euclidean Algorithm)



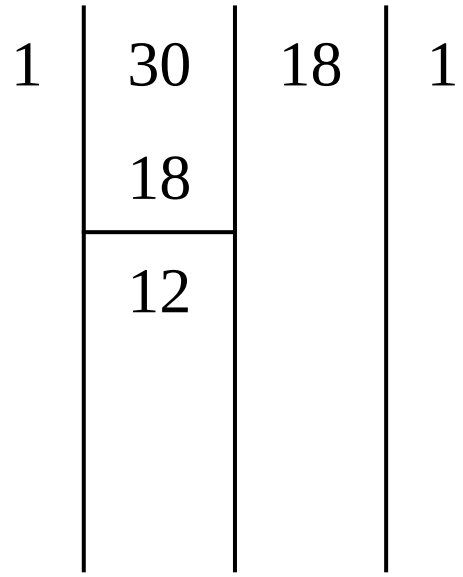
輾轉相除法 (Euclidean Algorithm)

1	30	18
	18	

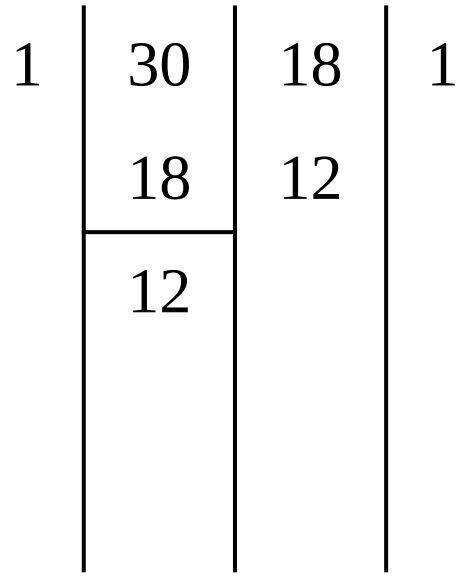
輾轉相除法 (Euclidean Algorithm)



輾轉相除法 (Euclidean Algorithm)



輾轉相除法 (Euclidean Algorithm)



輾轉相除法 (Euclidean Algorithm)

1	30	18	1
	18	12	
	12	6	

輾轉相除法 (Euclidean Algorithm)

1	30	18	1
	18	12	
2	12	6	

輾轉相除法 (Euclidean Algorithm)

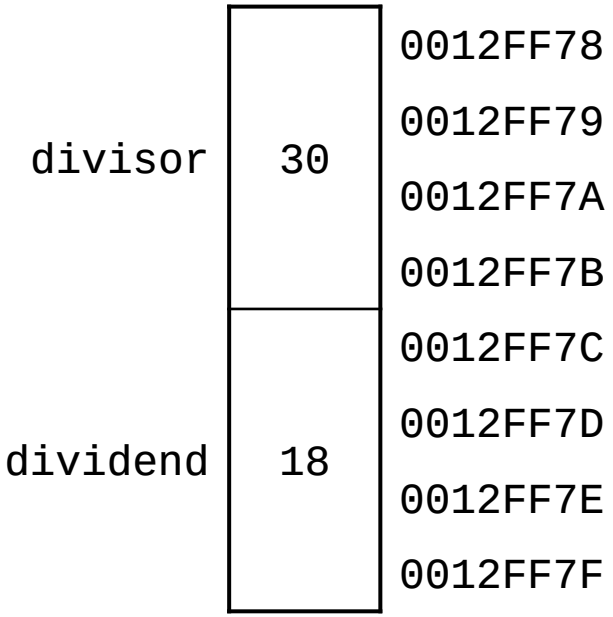
1	30	18	1
	18	12	
2	12	6	
	12		

輾轉相除法 (Euclidean Algorithm)

1	30	18	1
	18	12	
2	12	6	
	12		
	0		

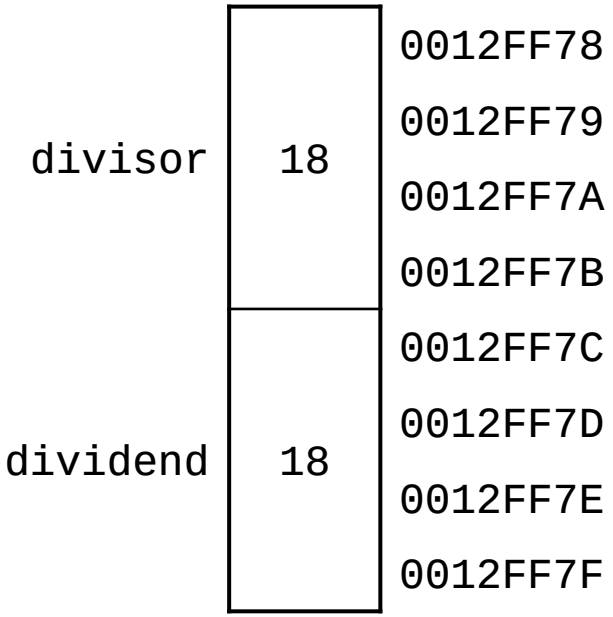
Greatest common divisor

```
if( divisor > dividend )
{
   divisor = dividend;
   dividend = divisor;
}
```



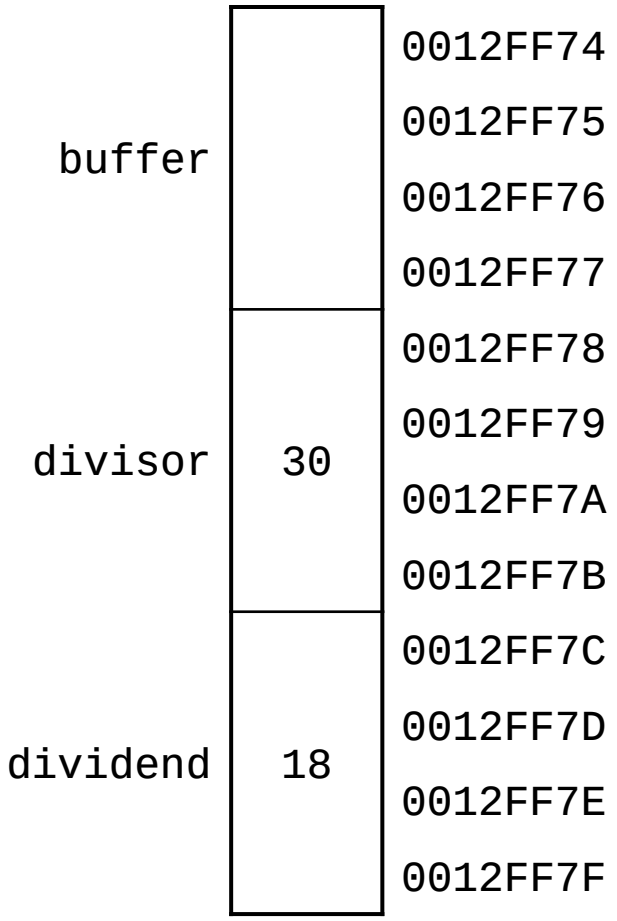
Greatest common divisor

```
if( divisor > dividend )
{
   divisor = dividend;
   dividend = divisor;
}
```



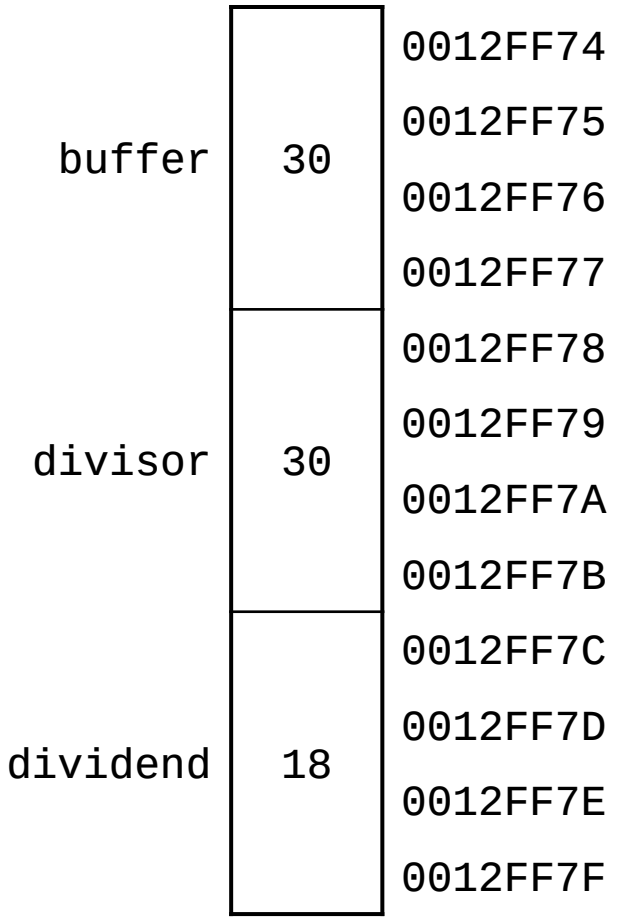
Greatest common divisor

```
if( divisor > dividend )
{
   buffer = divisor;
   divisor = dividend;
   dividend = buffer;
}
```



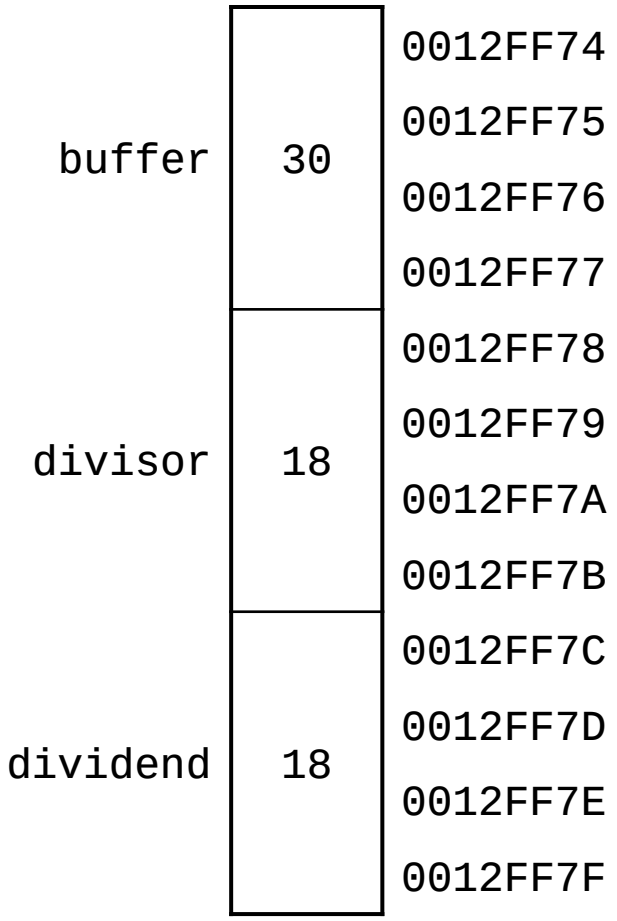
Greatest common divisor

```
if( divisor > dividend )
{
   buffer = divisor;
   divisor = dividend;
   dividend = buffer;
}
```



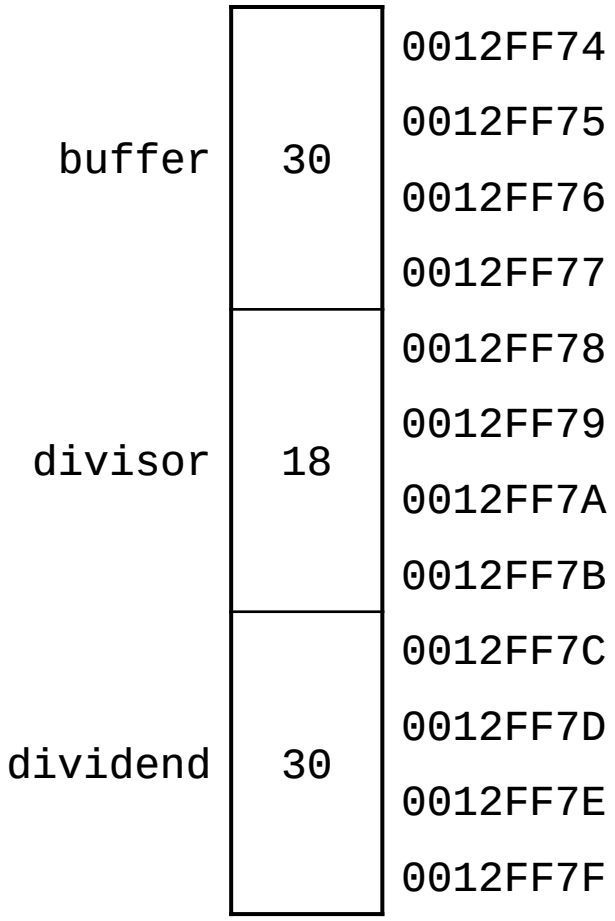
Greatest common divisor

```
if( divisor > dividend )
{
   buffer = divisor;
   divisor = dividend;
   dividend = buffer;
}
```

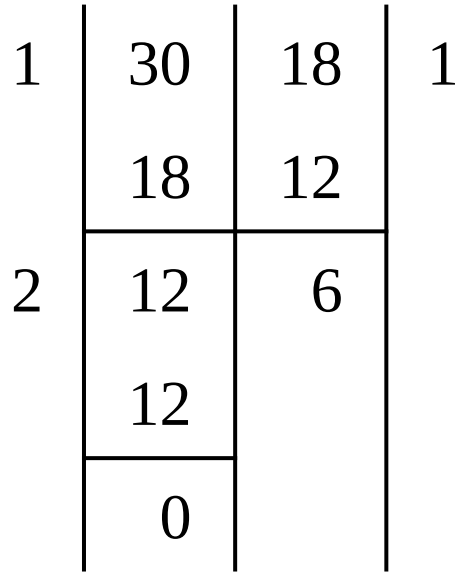


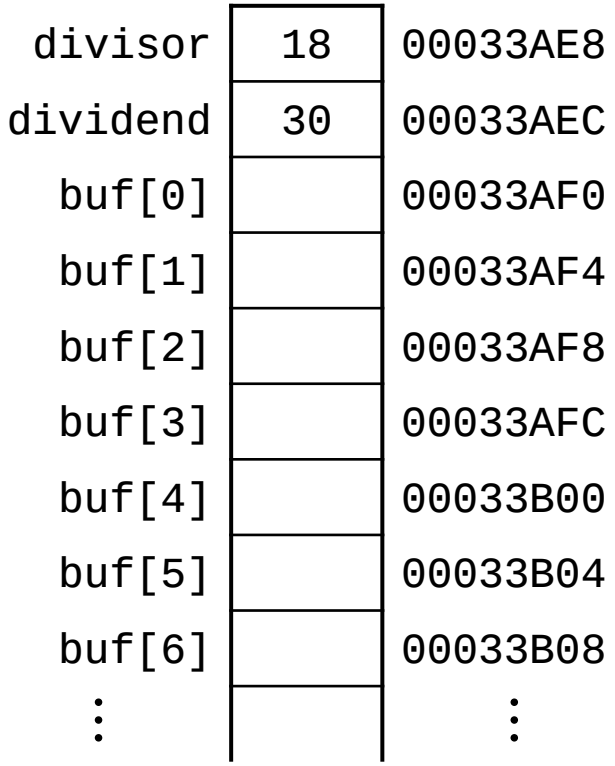
Greatest common divisor

```
if( divisor > dividend )
{
   buffer = divisor;
   divisor = dividend;
   dividend = buffer;
}
```



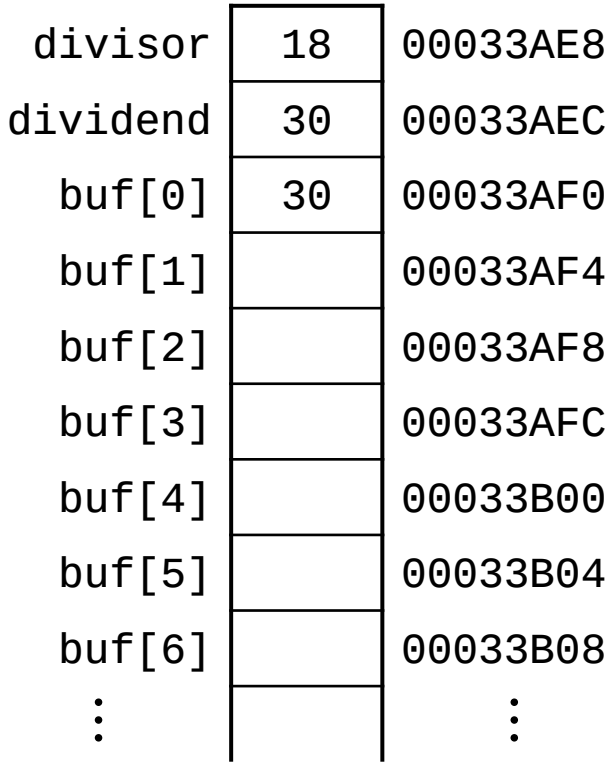
Greatest common divisor





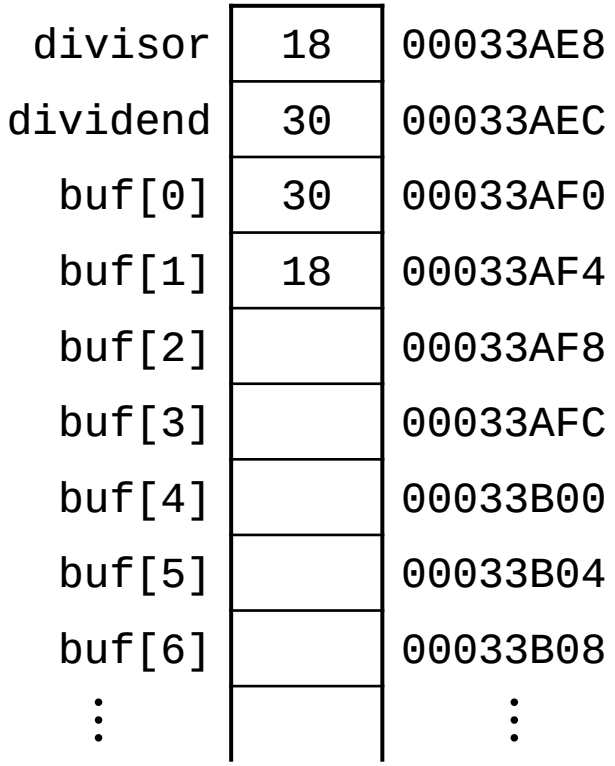
Greatest common divisor

1	30	18	1
	18	12	
2	12	6	
	12		
	0		

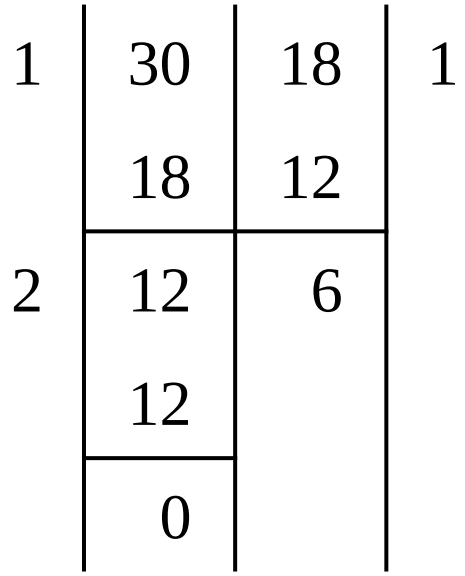


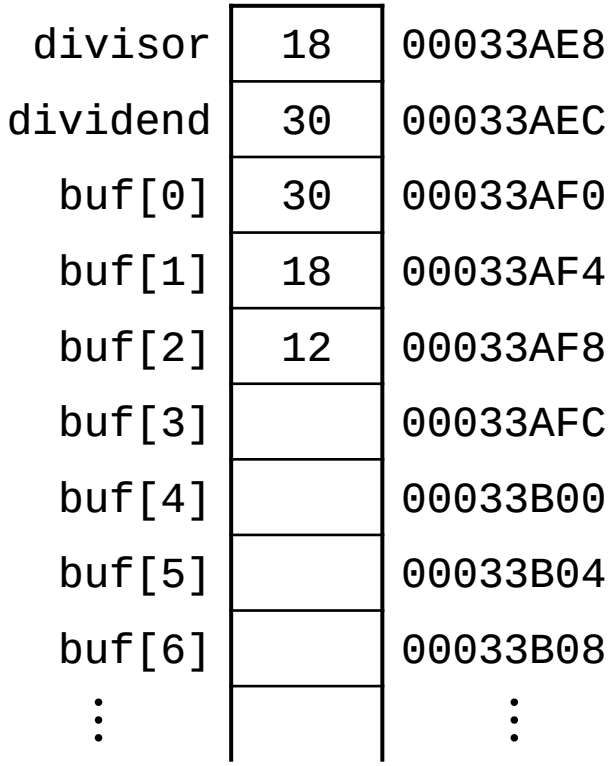
Greatest common divisor

1	30	18	1
	18	12	
2	12	6	
	12		
	0		



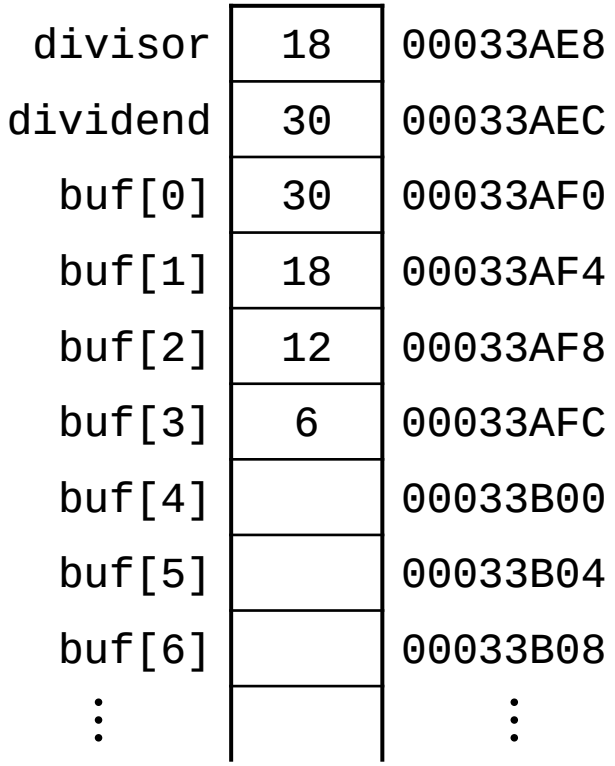
Greatest common divisor





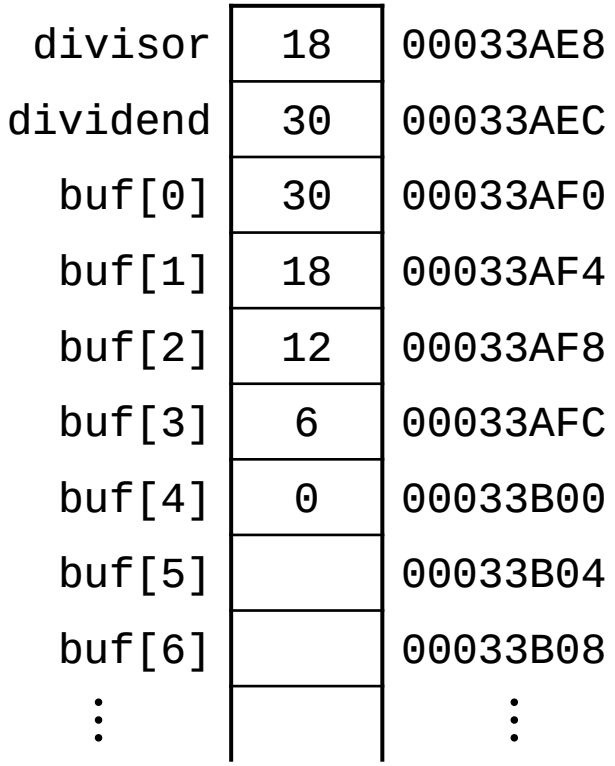
Greatest common divisor

1	30	18	1
	18	12	
2	12	6	
	12		
	0		



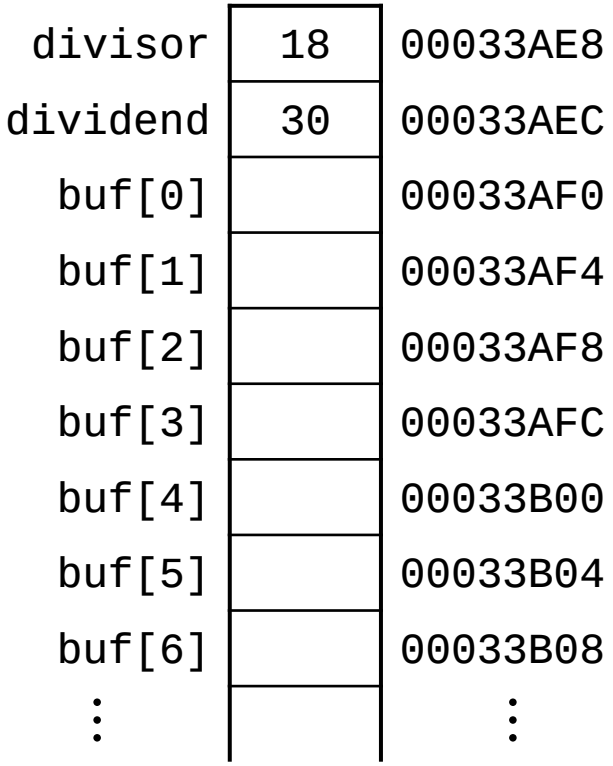
Greatest common divisor

1	30	18	1
	18	12	
2	12	6	
	12		
	0		



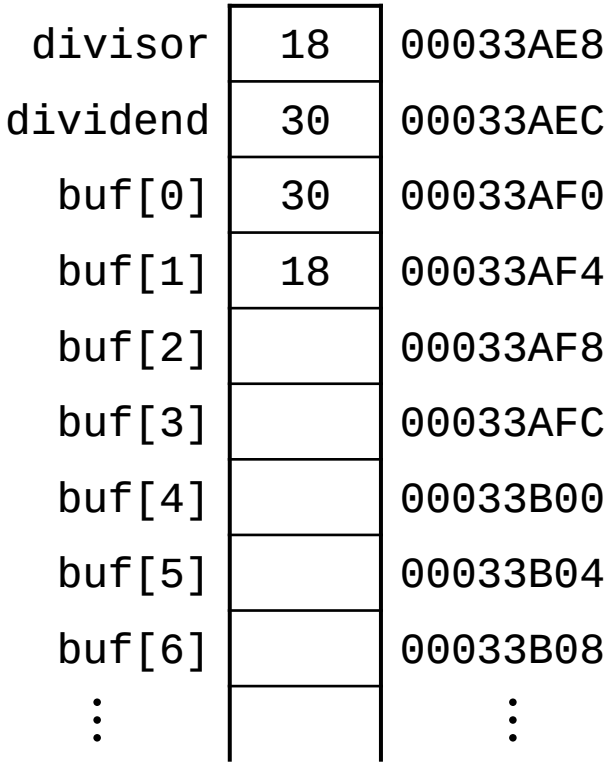
Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```



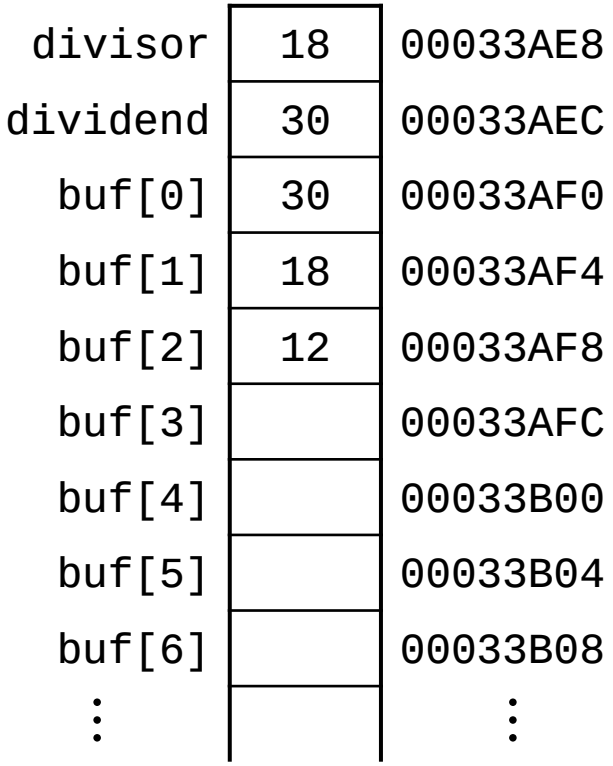
Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```



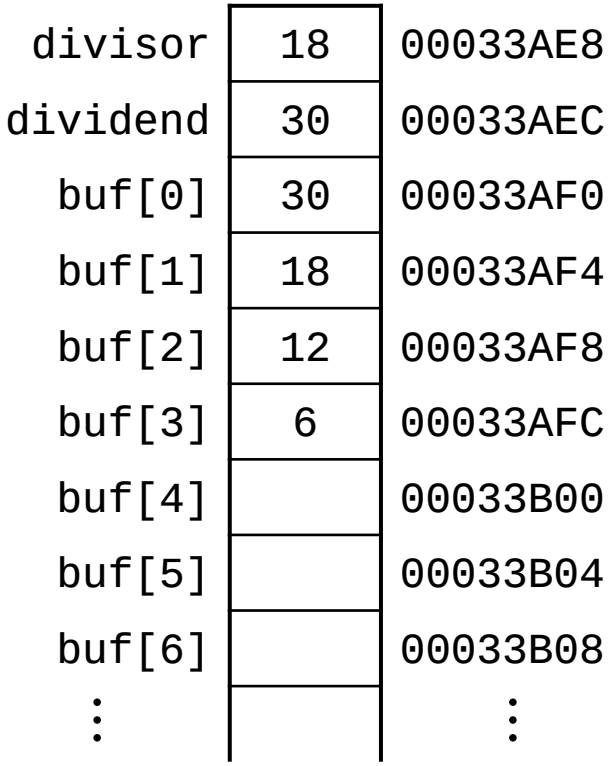
Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```



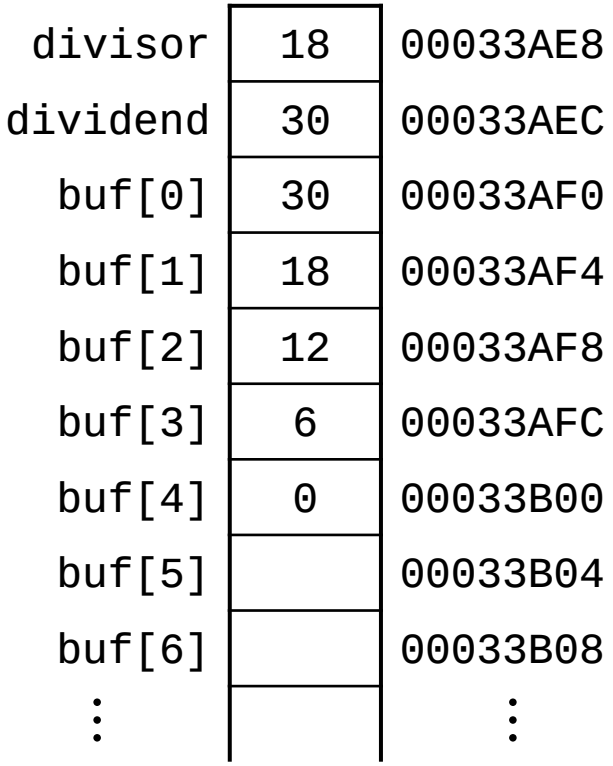
Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```



Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```



Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```

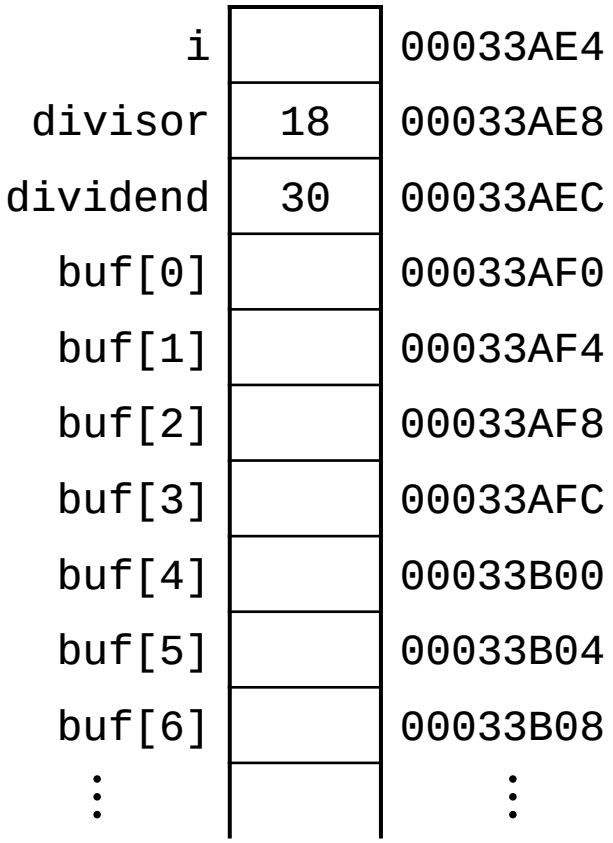
```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 };                != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
buf[ 2 ] = buf[ 0 ] % buf[ 1 ];
buf[ 3 ] = buf[ 1 ] % buf[ 2 ];
buf[ 4 ] = buf[ 2 ] % buf[ 3 ];
cout << buf[ 3 ] << endl;
```

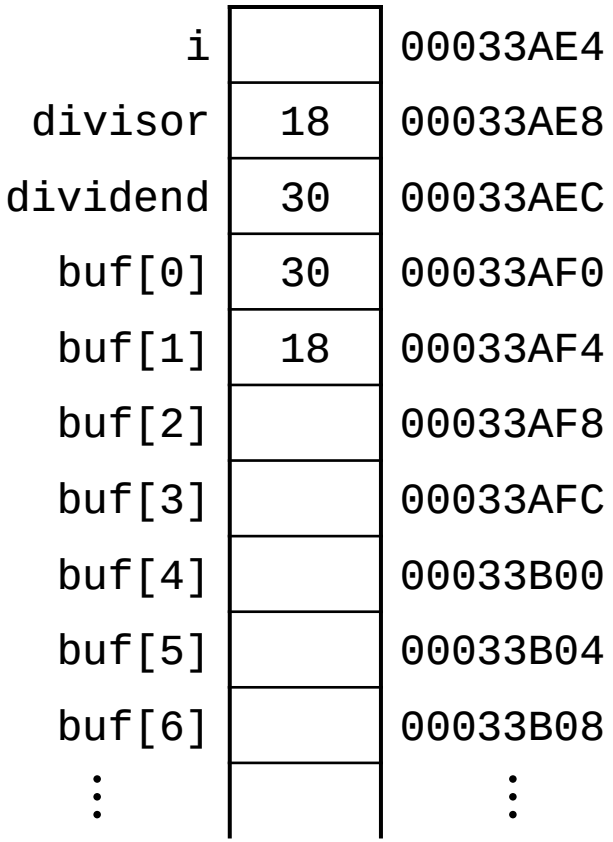
```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



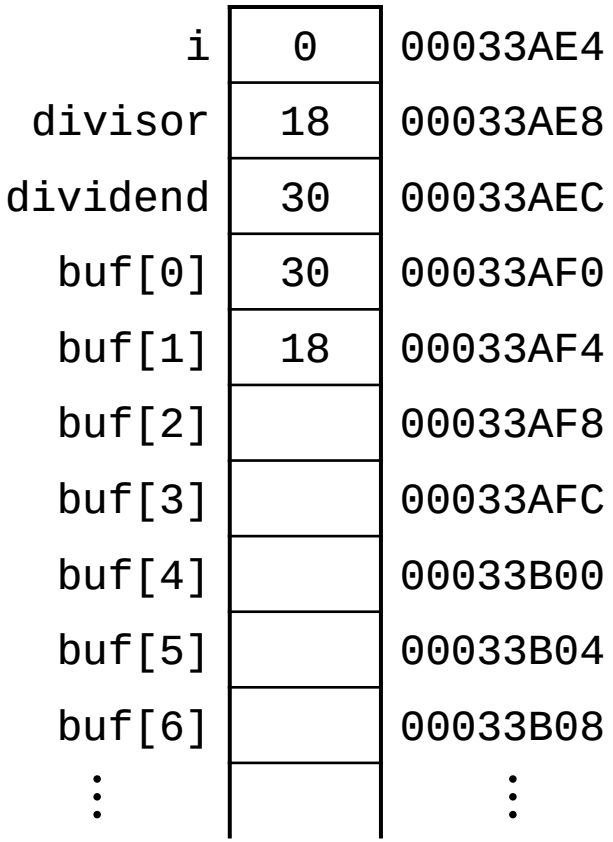

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



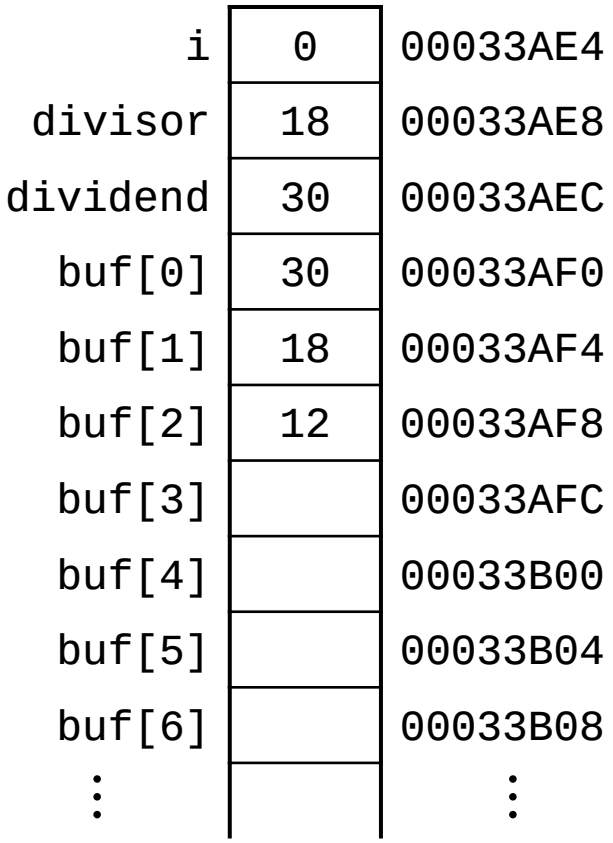

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



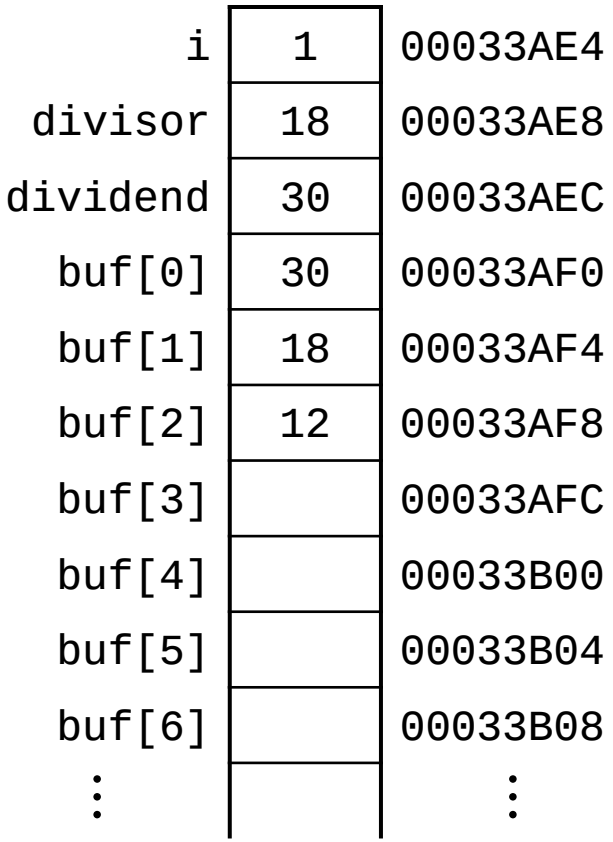

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



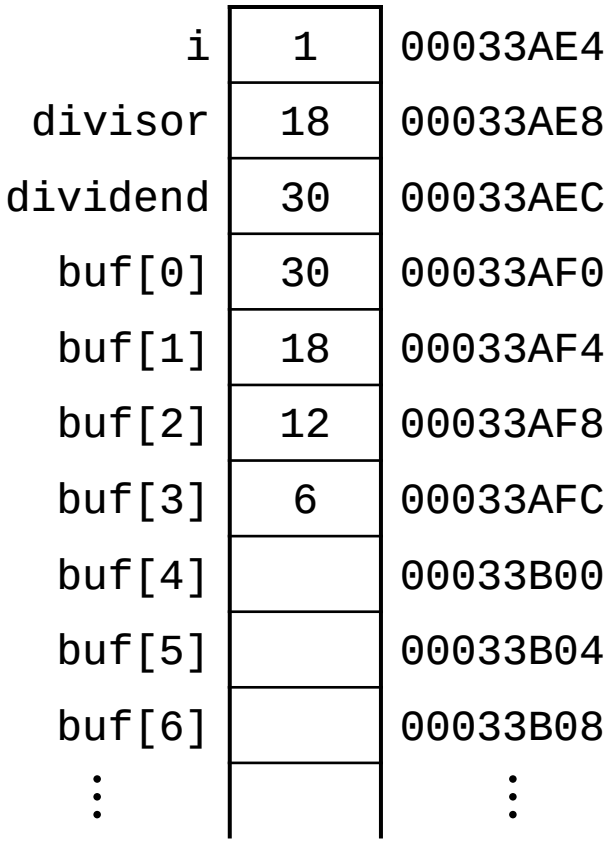

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



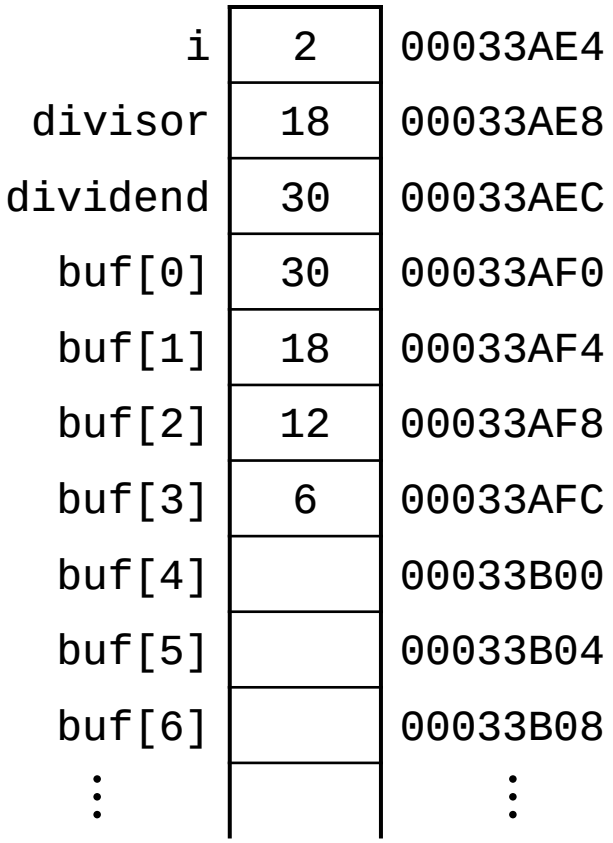

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



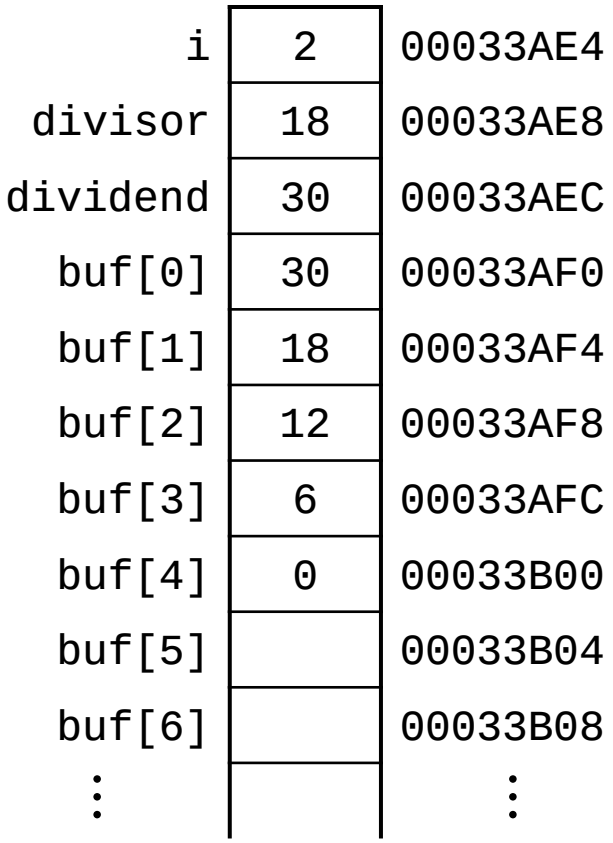

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



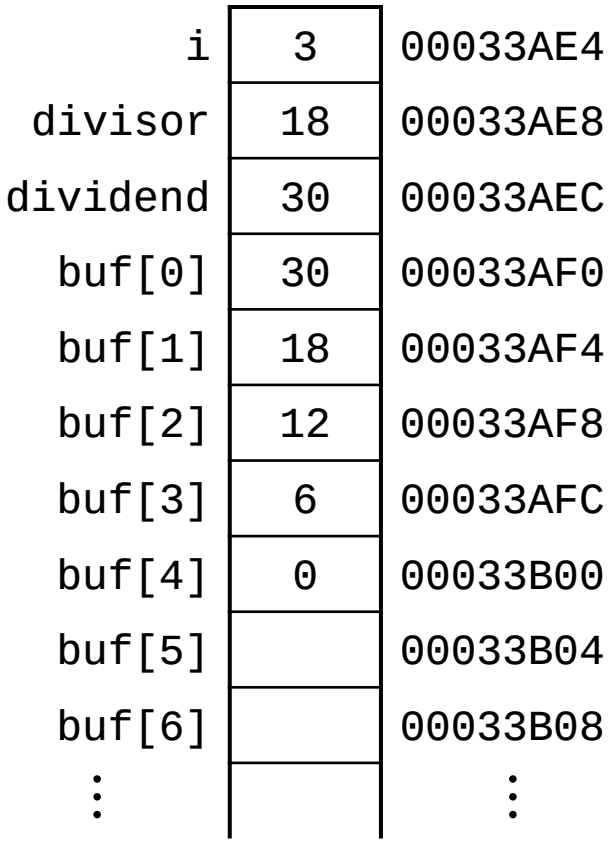

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



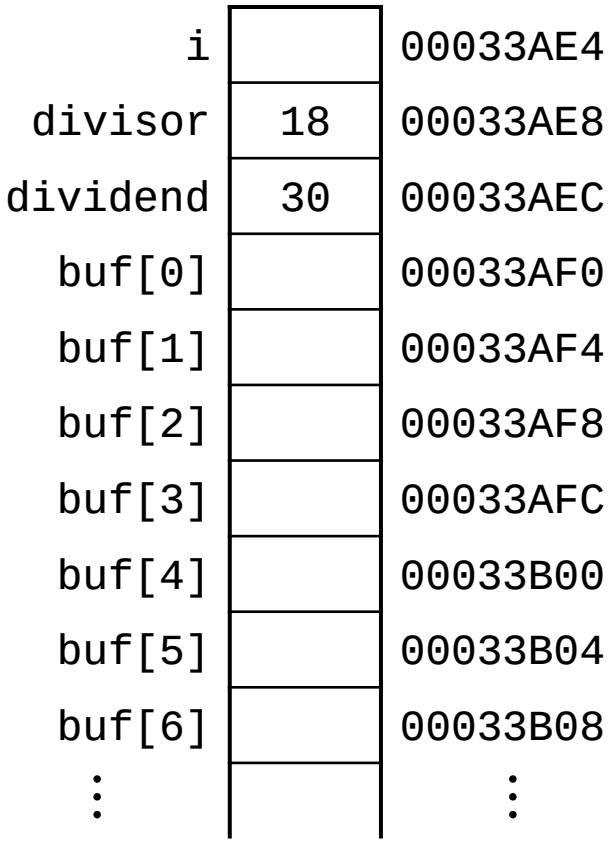

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



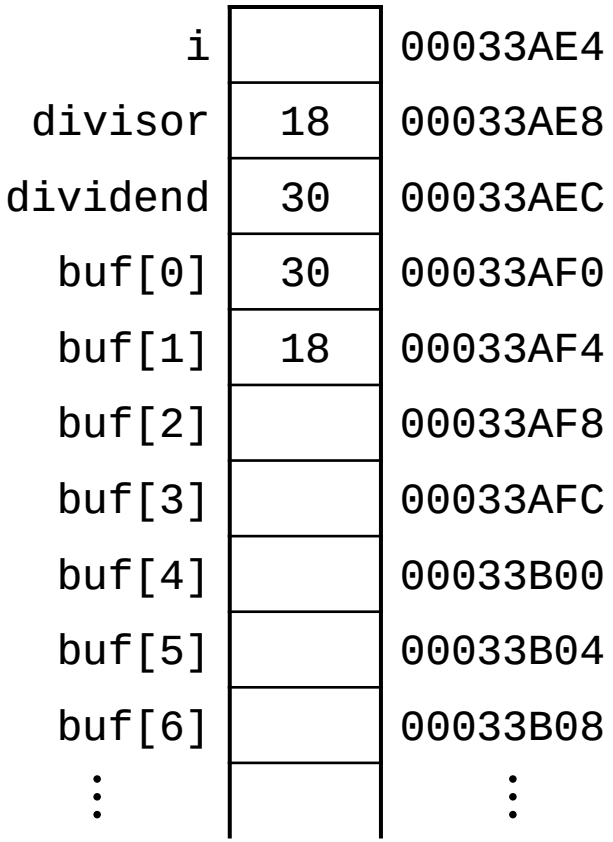

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[       ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[   ] << endl << endl;
```

Greatest common divisor



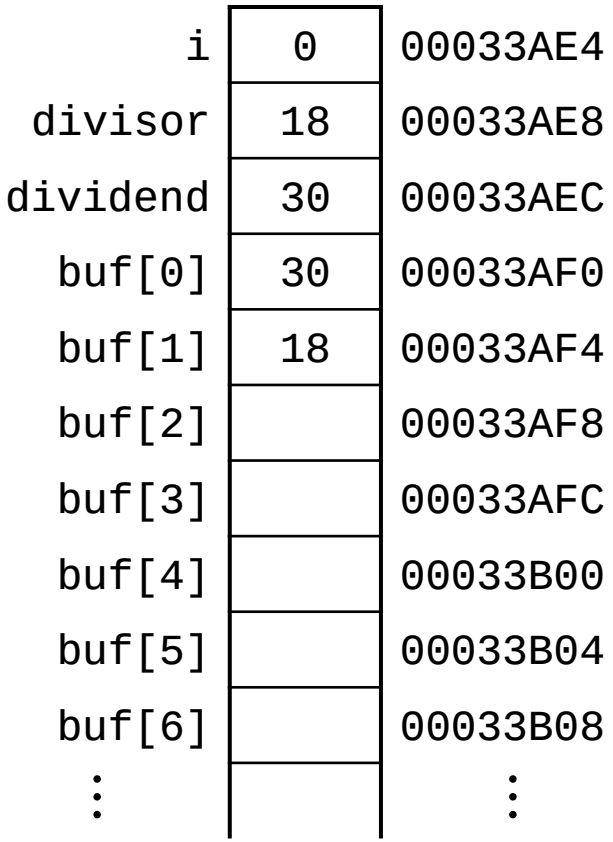

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor



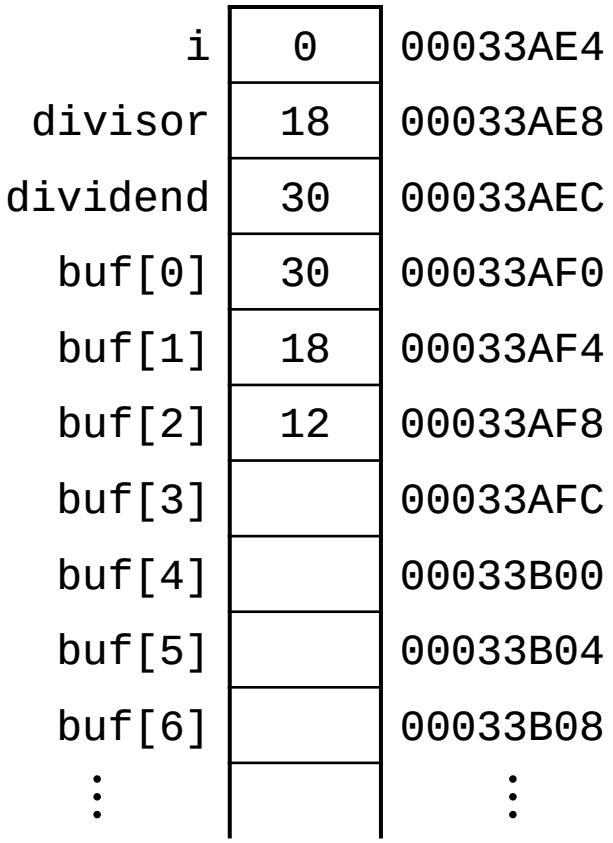

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor



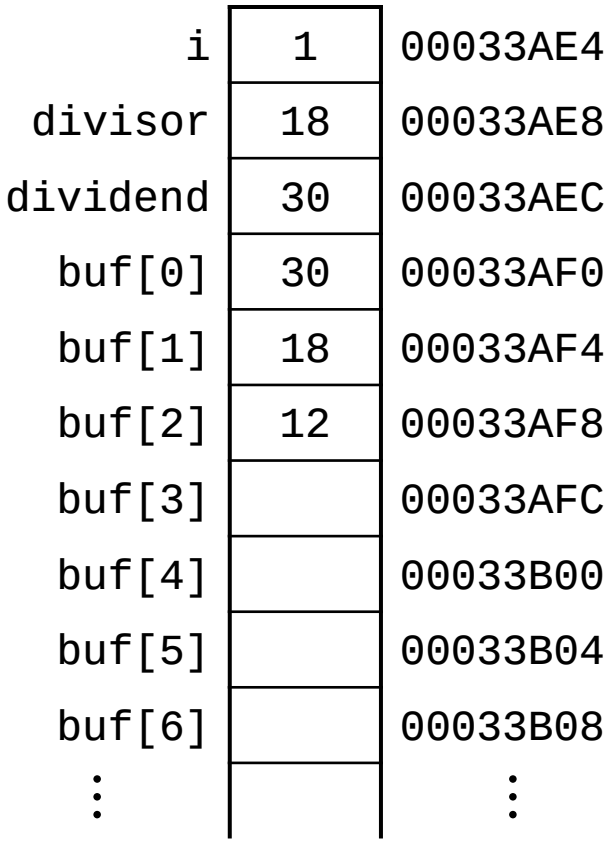

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor



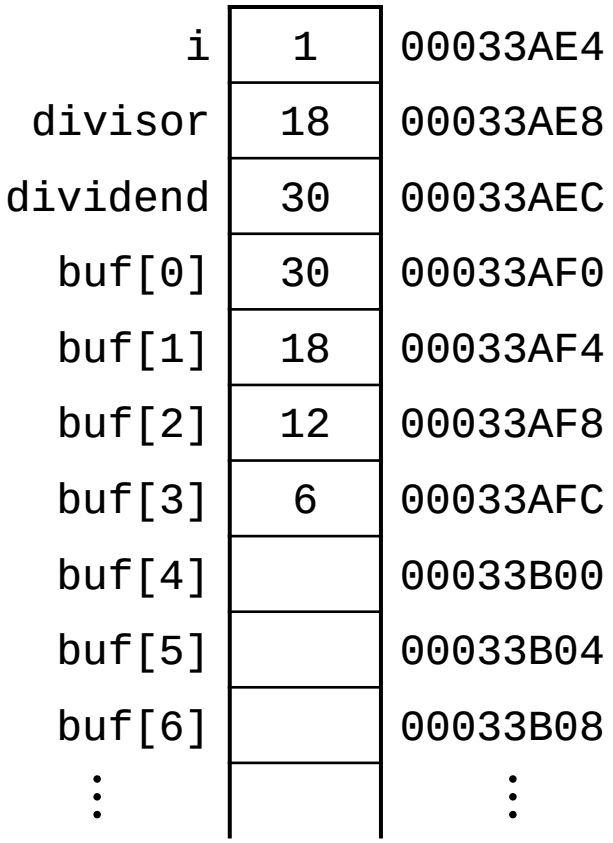

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor



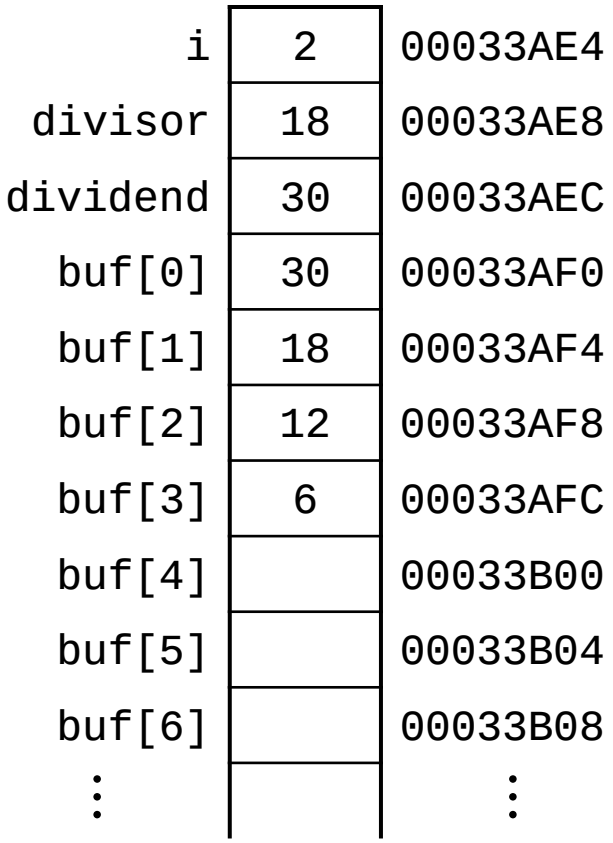

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor



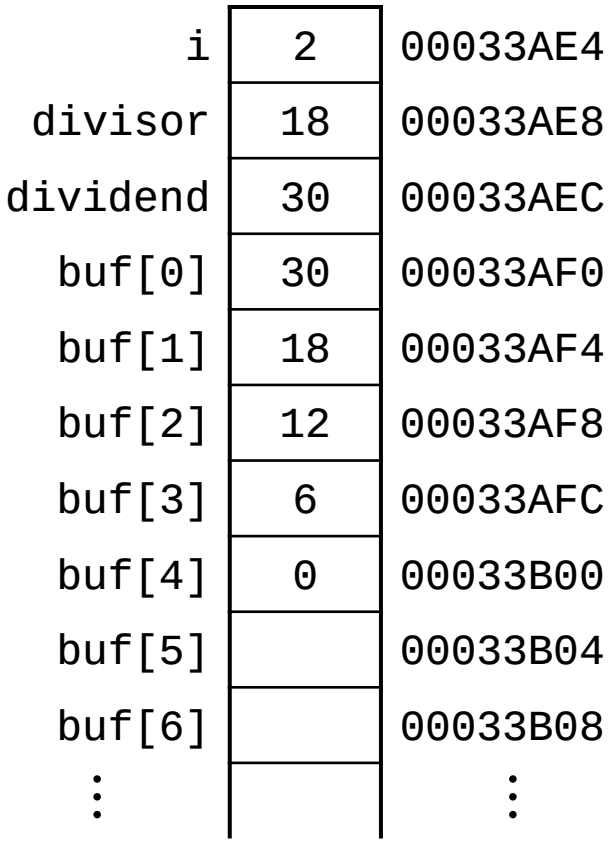

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor



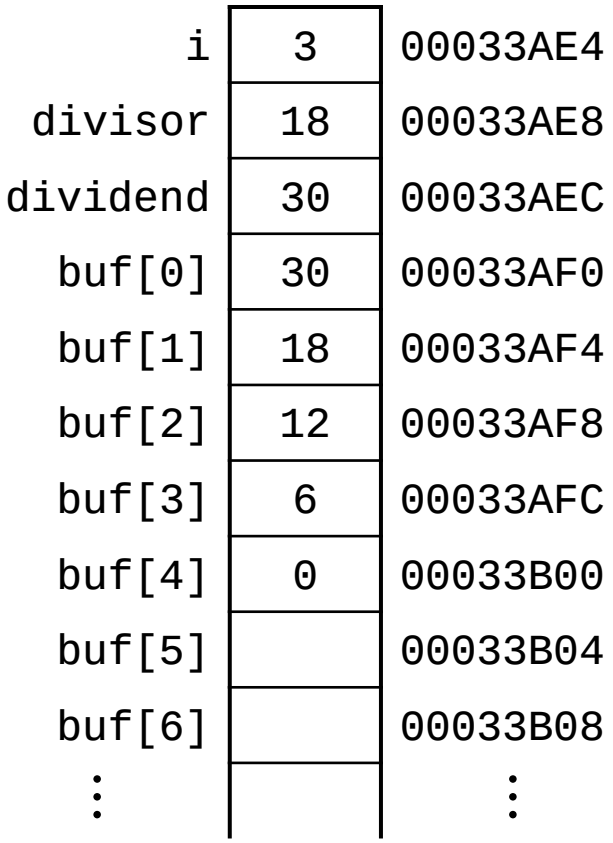

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor




```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

Greatest common divisor

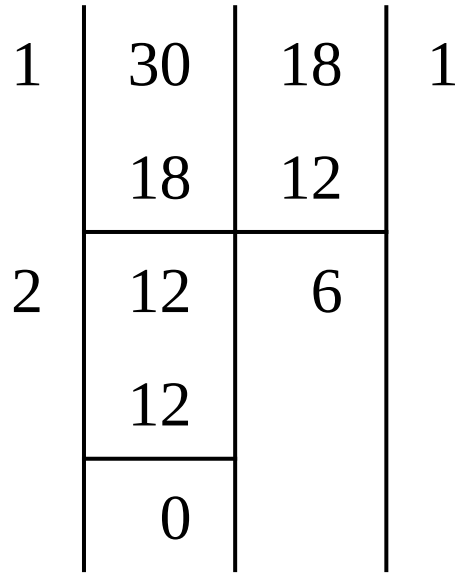


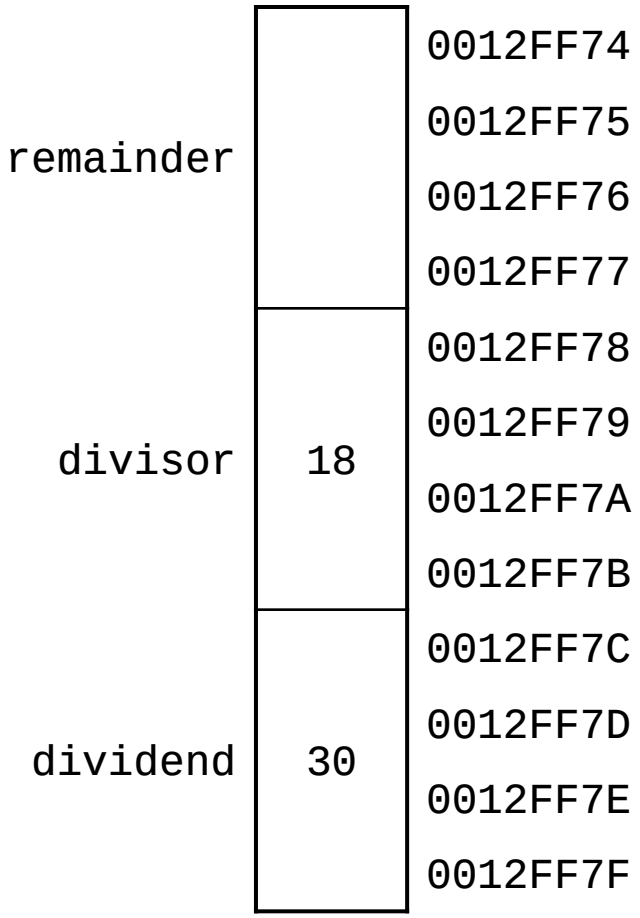

```
buf[ 0 ] = dividend;
buf[ 1 ] = divisor;
for( int i{ 0 }; buf[ i + 1 ] != 0; i++ )
   buf[ i + 2 ] = buf[ i ] % buf[ i + 1 ];
cout << buf[ i ] << endl << endl;
```

```
int main()
{
   int dividend, divisor;
   int hold;
   cin >> dividend >> divisor;

   if( divisor > dividend )
   {
      hold = divisor;
      divisor = dividend;
      dividend = hold;
   }

   int buffer[ 258340 ];
   buffer[ 0 ] = dividend;
   buffer[ 1 ] = divisor;

   for( int i{ 0 }; buffer[ i + 1 ] != 0; i++ )
      buffer[ i + 2 ] = buffer[ i ] % buffer[ i + 1 ];

   cout << buffer[ i ] << endl << endl;
}
```

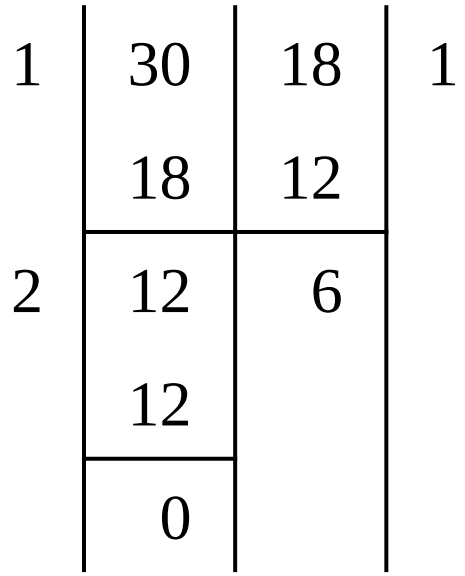
Don’t use arrays

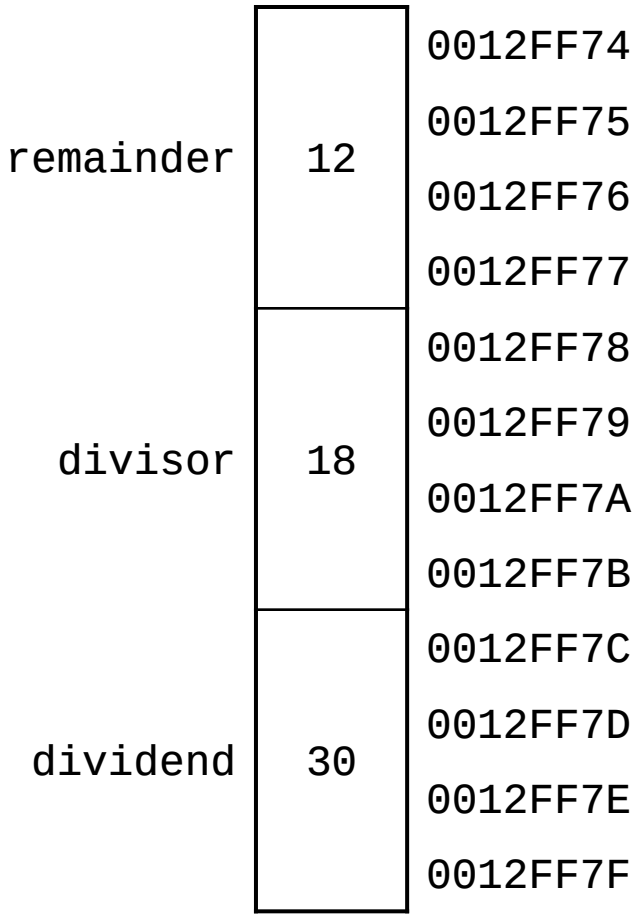
Greatest common divisor



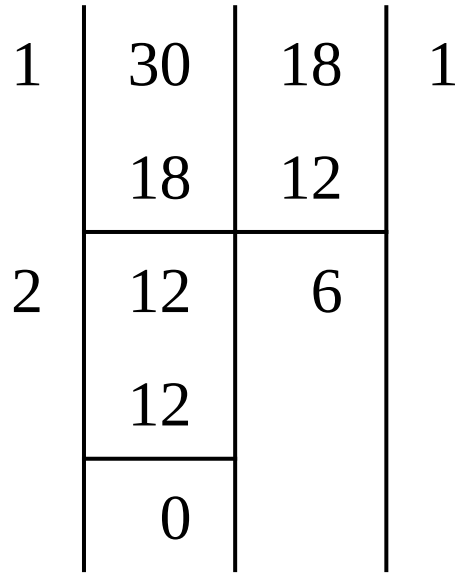


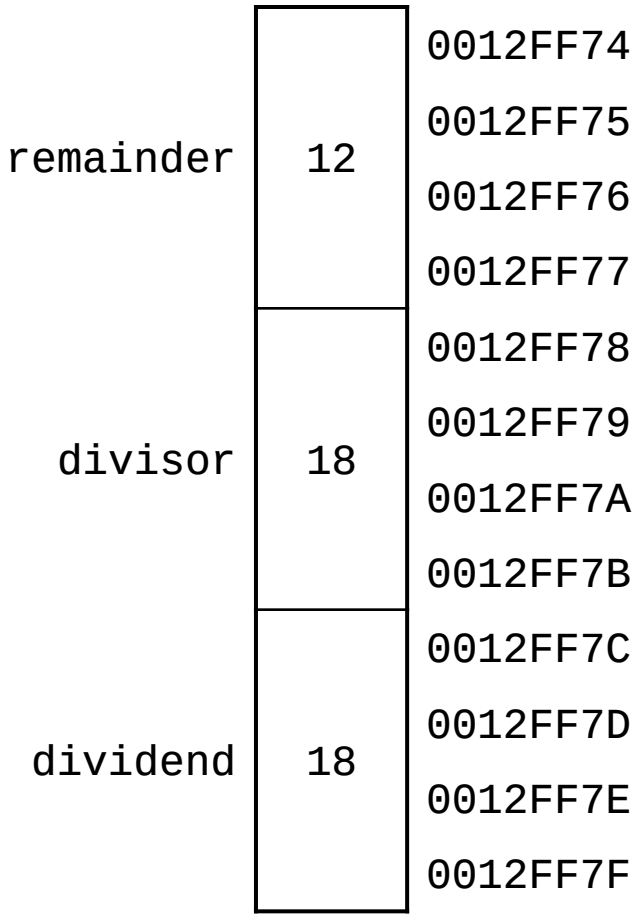
Greatest common divisor



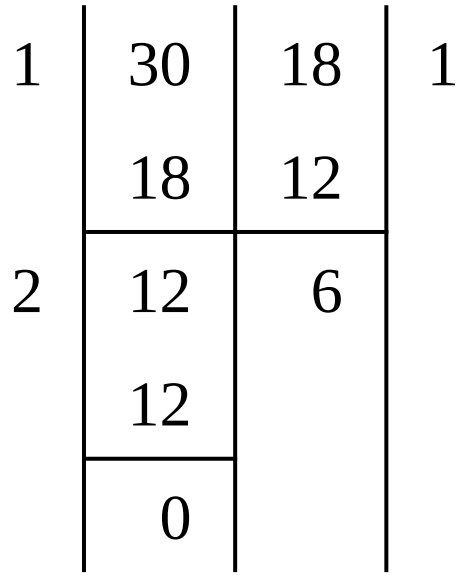


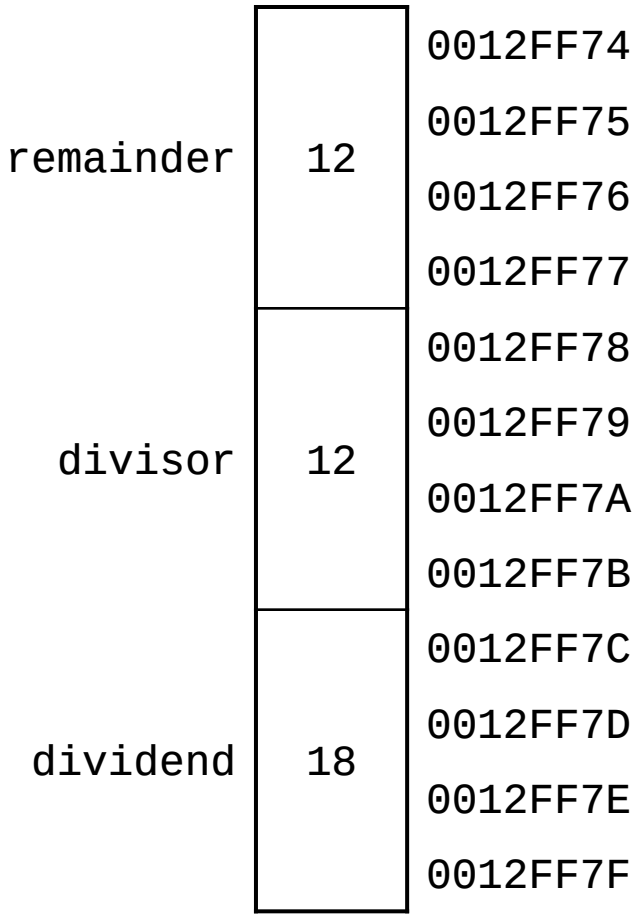
Greatest common divisor



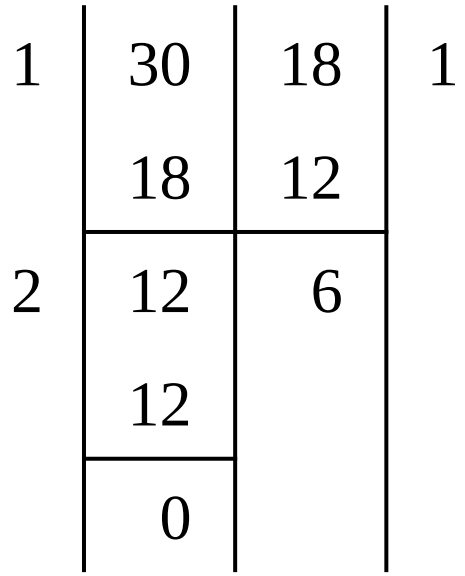


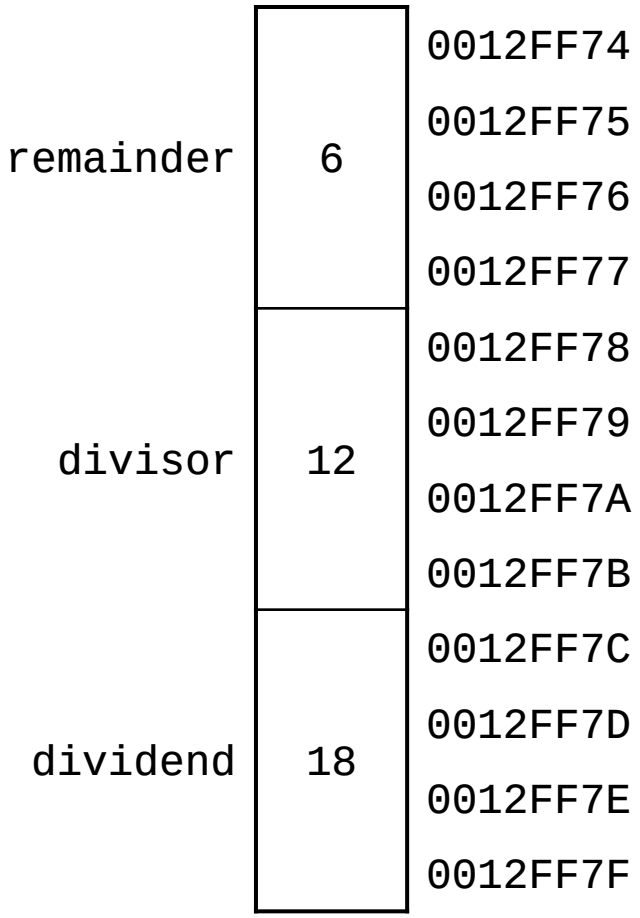
Greatest common divisor



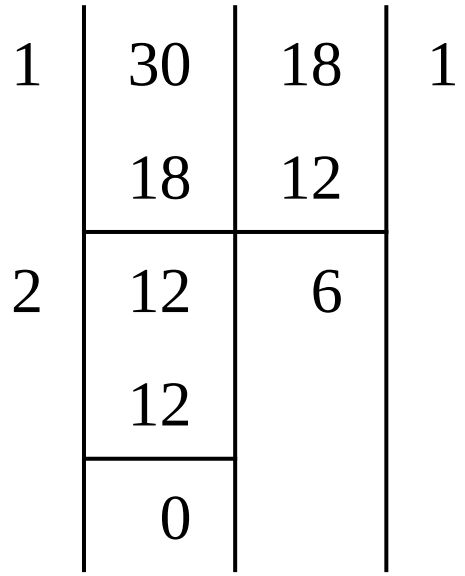


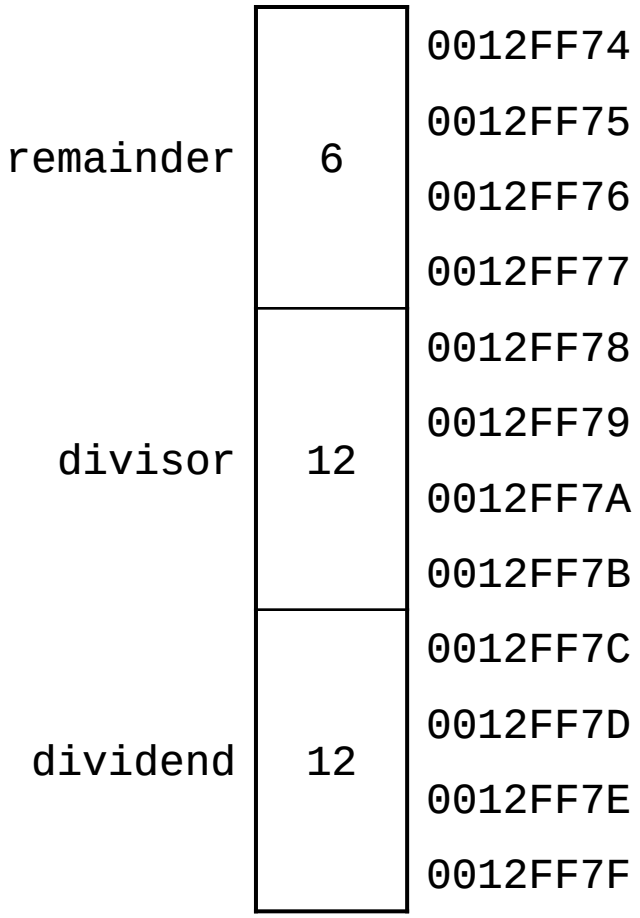
Greatest common divisor



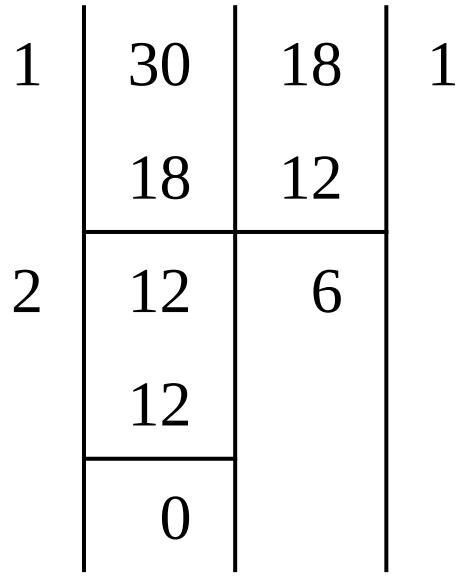


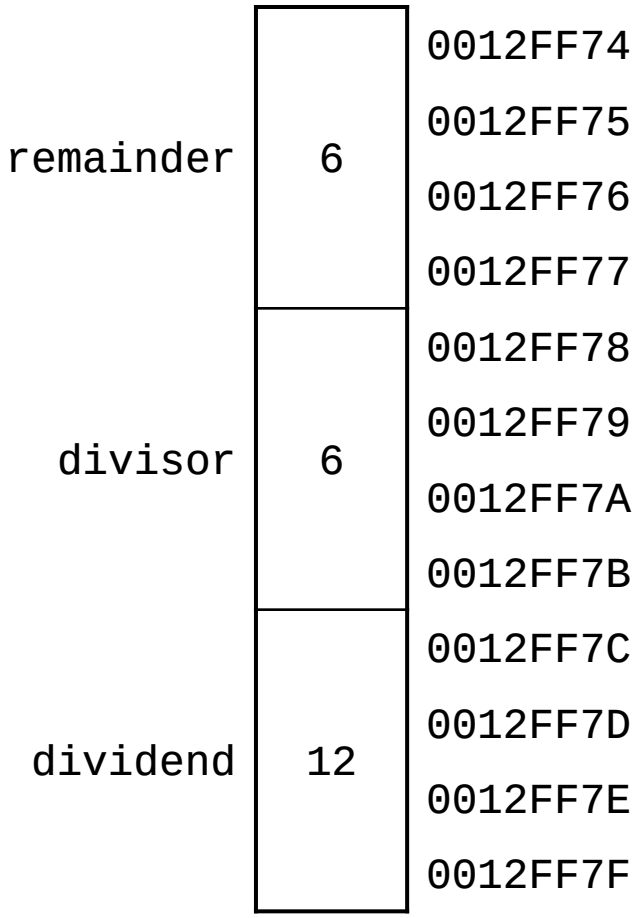
Greatest common divisor



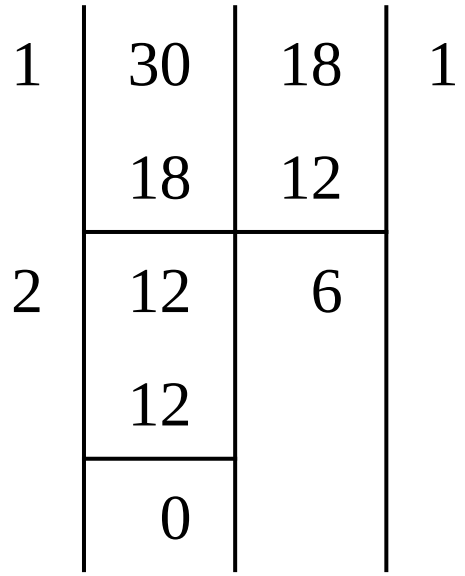


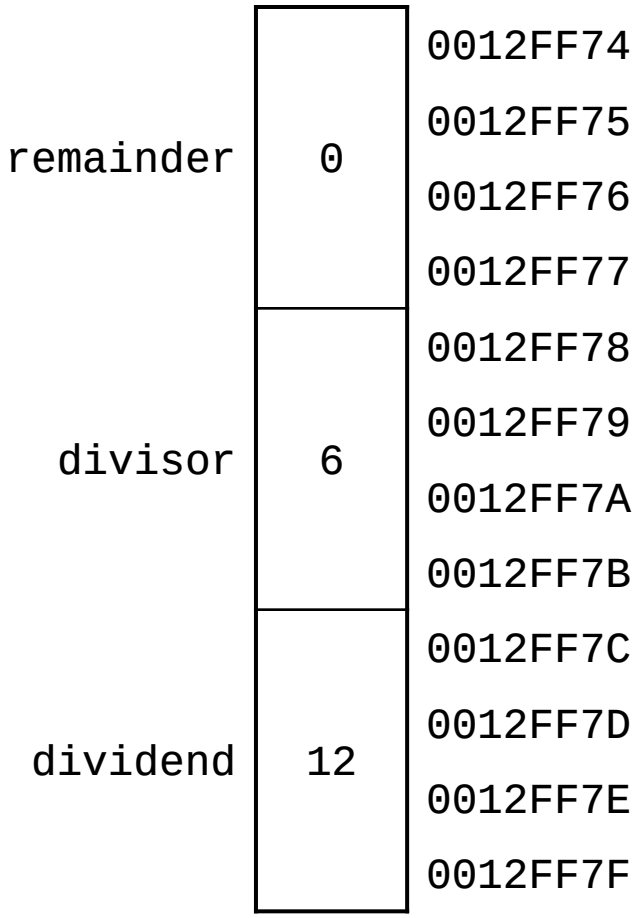
Greatest common divisor





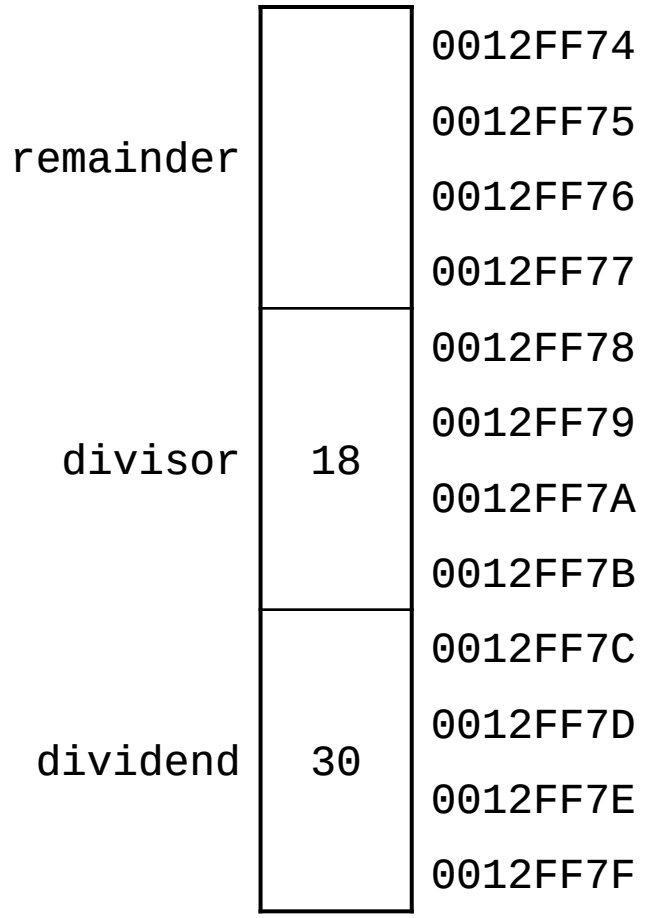
Greatest common divisor





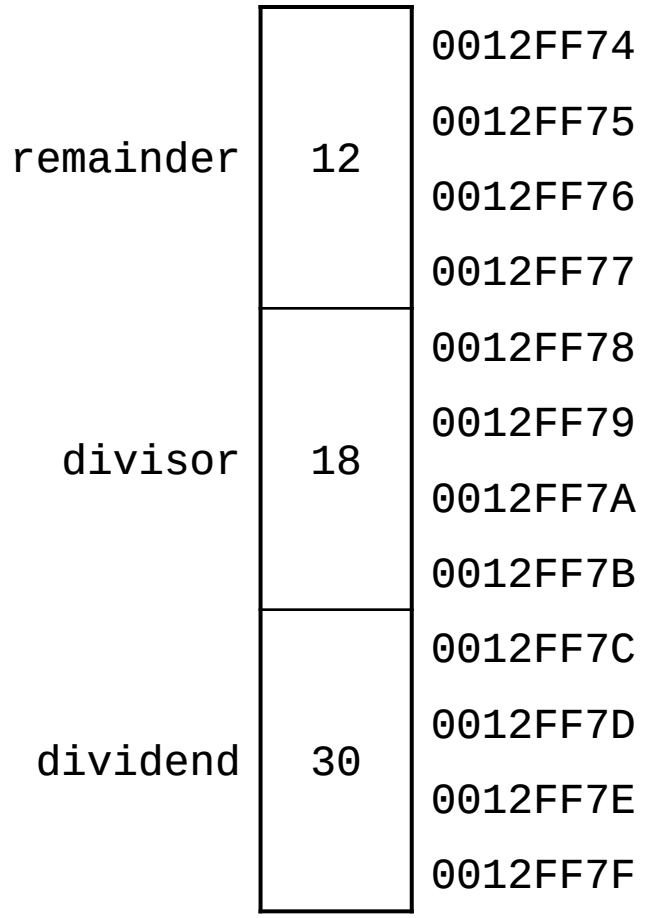
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



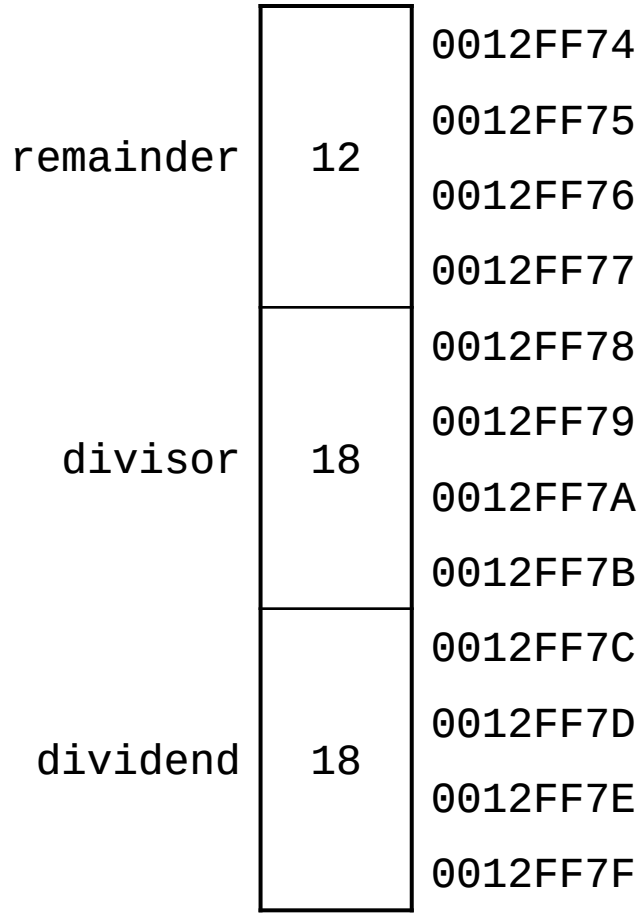
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



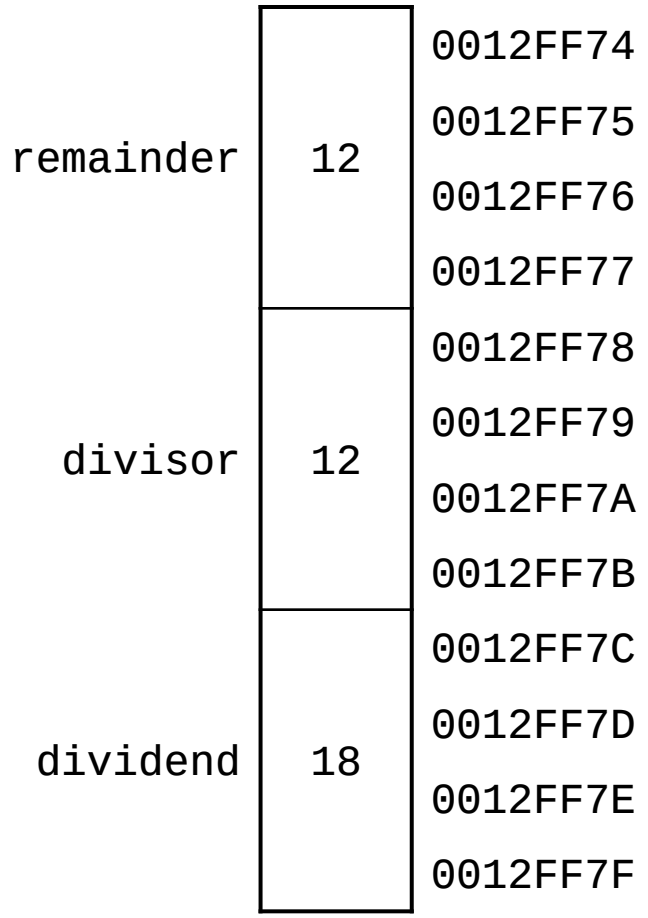
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



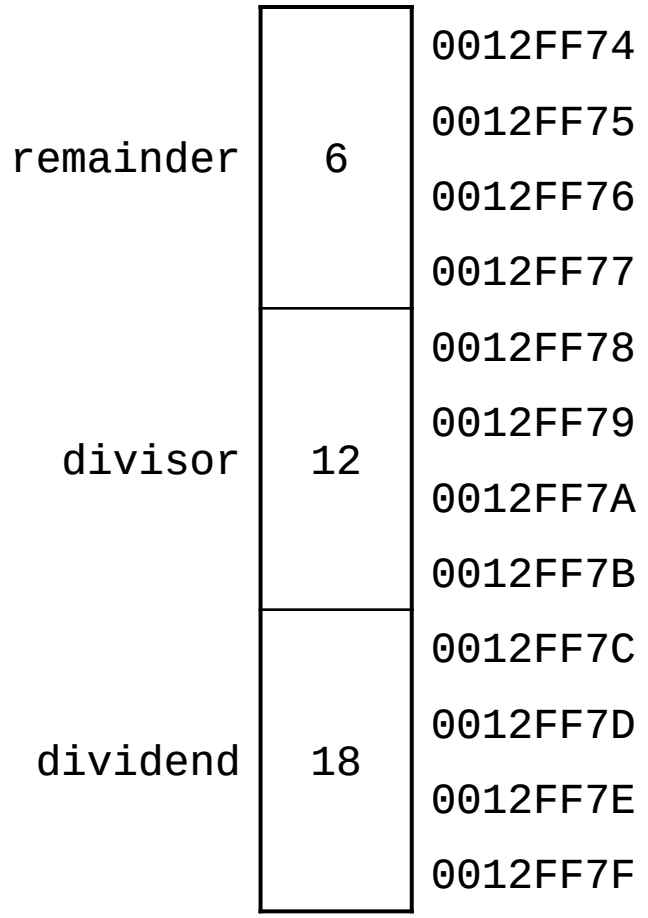
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



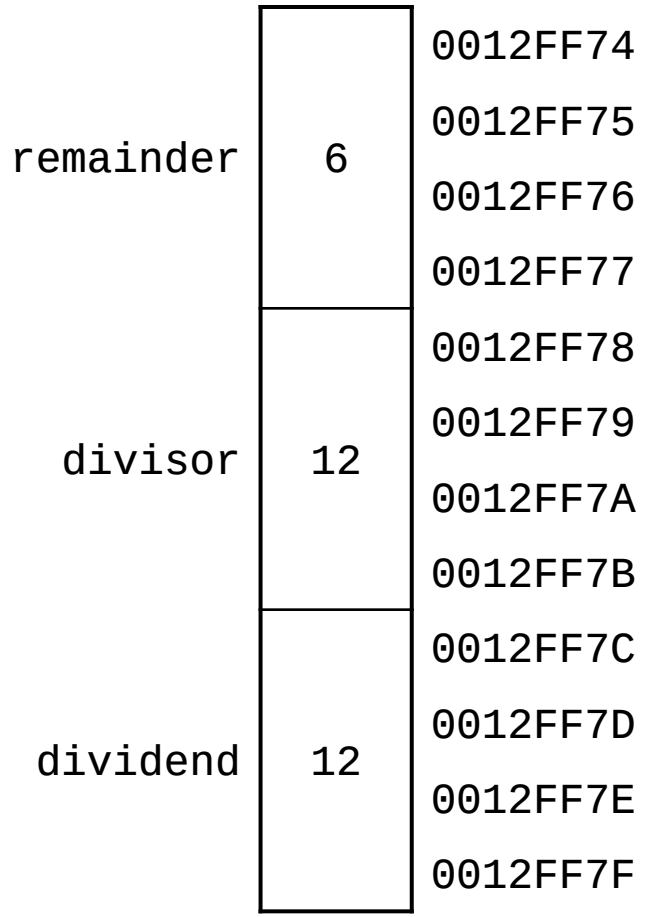
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



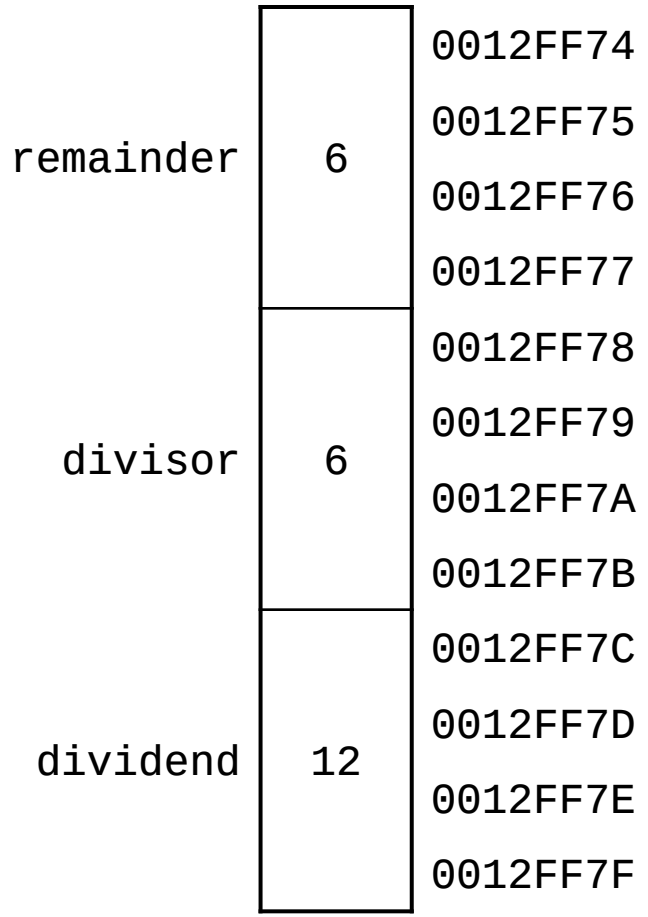
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



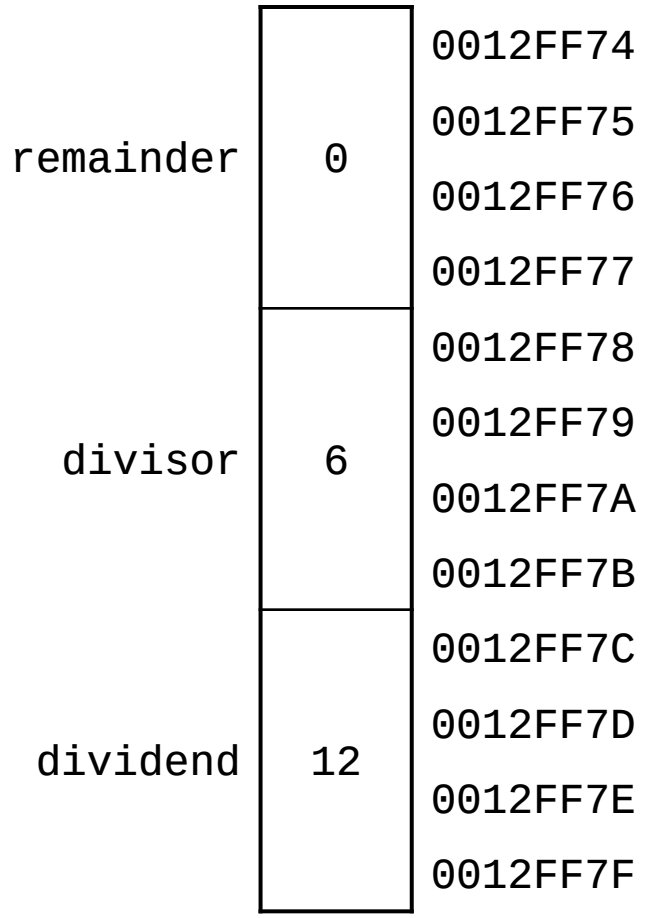
Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



Greatest common divisor

```
remainder = dividend % divisor;
while( remainder != 0 )
{
  dividend = divisor;
  divisor = remainder;
  remainder = dividend % divisor;
}
```



```
int main()
{
   int dividend, divisor, buffer;
   cin >> dividend >> divisor;
   if( divisor > dividend )
   {
      buffer = divisor;
      divisor = dividend;
      dividend = buffer;
   }

   int remainder = dividend % divisor;
   while( remainder != 0 )
   {
      dividend = divisor;
      divisor = remainder;
      remainder = dividend % divisor;
   }
   cout << divisor << endl << endl;
}
```